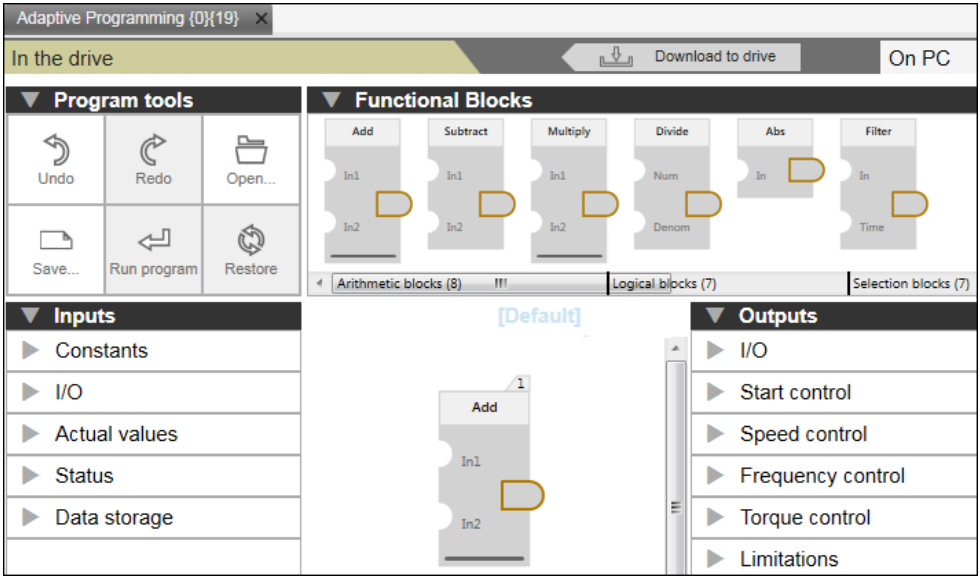
ABB industrial drives

Application guide
Adaptive Programming



Power and productivity
for a better world™

List of related manuals

Drive firmware manuals	Code (English)
ACS880 primary control program firmware manual	*3AUA0000085967*

Option manuals	
Drive composer start-up and maintenance PC tool User's manual	*3AUA0000094606*

You can find manuals and other product documents in PDF format on the Internet. See section *Document library on the Internet* on the inside of the back cover. For manuals not available in the Document library, contact your local ABB representative.

Application guide

Adaptive Programming

Table of contents

© 2015 ABB Oy. All Rights Reserved.

3AXD50000028574 Rev A
EN
EFFECTIVE: 2015-06-30

Table of contents

List of related manuals 2

1. Introduction to the guide

Contents of this chapter 7
Applicability 7
Compatibility 7
Drive composer pro 7
Drives 7
Safety instructions 8
Target audience 8
Purpose of the guide 8
Contents of the guide 8
Related documents 8

2. Adaptive Programming

Contents of this chapter 9
Overview of Adaptive Programming 9
Connecting a program to the drive application 10
Execution of a program 10

3. Using PC tool interface

Contents of this chapter 11
Adaptive programming user interface 11
Program tools 12
Function blocks 13
Inputs 14
Outputs 16
Creating a program 17
Downloading the program to a drive 19

4. Program elements

Contents of this chapter 23
System inputs 24
Constants 24
I/O 24
Actual values 24
Status 24
Data storage 24
System outputs 25
I/O 25
Start control 25
Speed control 25
Frequency control 26

Torque control 26
Limitations 26
Events 26
Process PID 26
Function block specifications 27
Abs 27
Add 28
AND 29
Divide 30
Equal 31
Filter 32
Greater than 33
Less than 34
Limit 35
Max 36
Min 37
Multiply 38
NOT 39
OR 40
PI 41
Ramp 43
Select boolean 45
Select value 47
Set bits 0-7 49
Set bits 8-15 51
SR 53
Subtract 54
Switch boolean 55
Switch value 57
Trigger down 59
Trigger up 60
T_off 61
T_on 62
XOR 63

Further information

Product and service inquiries 65
Product training 65
Providing feedback on ABB Drives manuals 65
Document library on the Internet 65

1

Introduction to the guide

Contents of this chapter

This chapter gives general information on the guide.

Applicability

This guide applies to the ACS880 control program and Drive composer pro (version 1.9 or later).

Compatibility

The guide complies with the drive application programs in which the Adaptive Programming features are included.

Drive composer pro

Adaptive Programming is available in Drive composer pro (version 1.9 or later) with Microsoft Windows 7 or newer. For further information on Drive composer pro, see the list of related manuals on the inside of the front cover.

Drives

Adaptive Programming is compatible with ACS880 primary control program version 2.20 or later.

Safety instructions

Follow all safety instructions delivered with the drive.

- Read the complete safety instructions before you install, commission, or use the drive. The complete safety instructions are delivered with the drive as either part of the Hardware manual, or, in the case of ACS880 multidrives, as a separate document.
- Read the software function specific warnings and notes before changing the default settings of the function. For each function, the warnings and notes are given in the Firmware Manual in the subsection describing the related user adjustable parameters.

Target audience

This guide is intended for people who design, commission, or operate the drive system.

Purpose of the guide

This guide is used together with the firmware manual of the drive application program.The firmware manual contains the basic information on the drive parameters including the parameters of the Adaptive Programming. This guide gives more detailed information on the Adaptive Programming:

-what is Adaptive Programming

-how to build a program

-how the function blocks operate

-how to use the system inputs and outputs

Contents of the guide

This guide contains the following chapters:

Adaptive Programming contains the overview of the Adaptive Programming.

Using PC tool interface provides the main UI elements of PC tool interface for Adaptive Programming.

Program elements contains descriptions of function blocks of Adaptive Programming.

Related documents

A list of related manuals are printed on the inside of the front cover.

2

Adaptive Programming

Contents of this chapter

This chapter describes the overview of the adaptive programming feature.

Overview of Adaptive Programming

Adaptive programming is used to customize the operation of a drive in case the drive parameter setting is not sufficient. The program is built with standard function blocks included in the drive firmware. A predefined list of inputs is available for getting information from the drive parameters to use in the adaptive program. A predefined list of outputs is available that defines parameters where it is possible to write from the adaptive program.

Adaptive program is created with Drive composer pro. Drive composer pro is used to download adaptive program to the drive and to start the adaptive program. By default, adaptive program is started when the drive is powered *ON* if the adaptive program exists in the drive.

Standard function blocks (for example Add, AND) are used to create an executable adaptive program. The maximum size of an adaptive program is approximately 20 standard function blocks, depending on the block types used and the number of predefined inputs and outputs utilized in the program. The standard function blocks available are presented in *Program elements* **(**page *23*). Numerical function blocks use floating point numbers in the calculations.

Connecting a program to the drive application

Adaptive program is connected to the drive application through predefined system inputs and outputs. Drive provides the available inputs and outputs and sets the pointer parameter values accordingly based on the created program.

When the predefined output (value/bit pointer parameter) is written to from the adaptive program, the parameter is write protected and it is not changed in the parameter table. The control panel and Drive composer pro shows a text in the pointer parameter to indicate that the parameter is connected to the adaptive program.

Execution of a program

Adaptive program is executed on firmware time level. Status of the adaptive program is seen from parameter *7.30 Adaptive program status*. If Adaptive program in the drive is incompatible or corrupted, the fault *64A6h Adaptive program* is activated. The extension code of the fault explains the detail of the problem with the adaptive program.

Note: For time level actual value, refer *firmware manual(s)* in *List of related manuals*.

Editing of the Adaptive program is possible only when the drive is in stopped state. While editing Adaptive program, the start inhibit is *ON*, so that the drive cannot be started.

The Adaptive program executes the function blocks in numerical order with all blocks on the same time level. This cannot be changed by the user.

The user can:

- build a program using blocks and connections

- change the numbering of the blocks by moving them to different positions

- select the operation mode of the program (run/edit).

3

Using PC tool interface

Contents of this chapter

This chapter describes the main UI elements of PC tool interface for adaptive programming.

Adaptive programming user interface

The main user interface of adaptive programming consists of the following sections:

- Program tools
- Function blocks
- Inputs
- Outputs

The working area can be used either with tab or floating window. The selection between tab and floating window can be made using Drive composer pro **View** menu. The figure below shows the user interface with tabbed window.

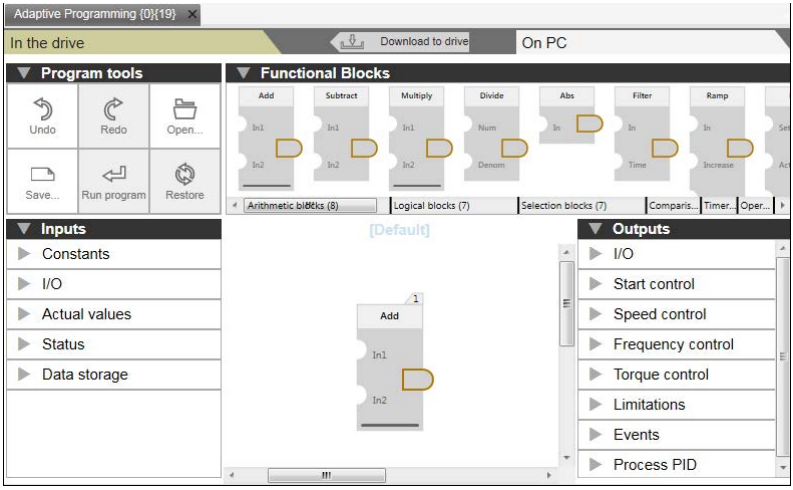


Figure 1. Adaptive programming

Program tools

The program tools contains the following options:

- Undo: Erases the last change made and reverts it to an older state.
- Redo: Reverses the undo or advances to a more current state.
- Open: Opens a program from locally saved file.
- Save: Saves the active program to a local file (*.dcap* format).
- Run program: Starts the program.
- Restore: Restores the default program.
- Import blocks: Allows to import function blocks from the file system.

Function blocks

Function blocks of adaptive programming are grouped into categories and are shown on a horizontal shelf. The scroll bar shows category labels and indicates the current view. The blocks are quickly accessible. The user can drag and drop the required blocks to the canvas. See *Adaptive programming* (page *12*).

Following are the function block categories:

- Arithmetic blocks
- Logical blocks
- Selection blocks
- Comparison blocks
- Timer blocks
- Operation blocks

Inputs

The predefined inputs are categorized into groups. The available groups and inputs are dependent on the drive type. Typical examples are:

- Constants
- I/O
- Actual values

You can use the same input multiple times in the same program. Hovering over an input on the shelf highlights every instance of that input on the canvas, so you can easily locate where the input is used in the program.

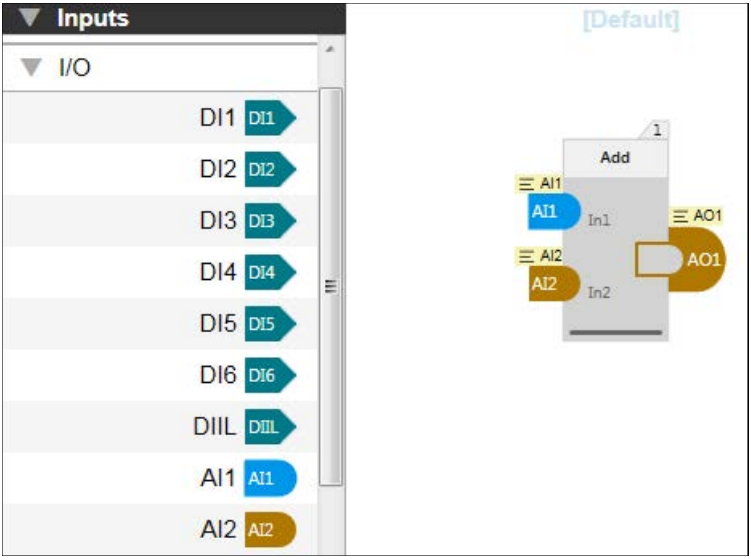


Figure 2. Inputs

To edit the input label and add a comment, proceed as follows:

1. Click DI label in the function block input.

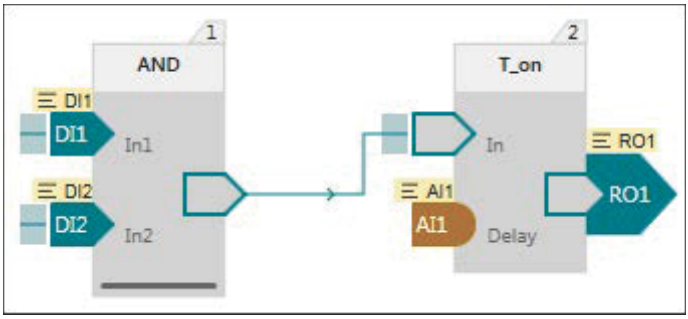


Figure 3. Editing label

2. Edit the label and add the comment as desired.

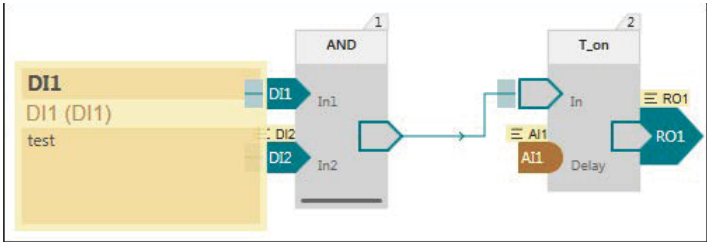


Figure 4. Editing label and comment

For more information on Input descriptions, refer *firmware manual(s)* in *List of related manuals*.

Outputs

The predefined outputs are categorized into groups. The available groups and outputs are dependent on the drive type. Typical examples are:

- I/O
- Start control
- Speed control

Each output can be used only once in the program. When you drag and drop an output to the canvas, it is faded on the shelf.

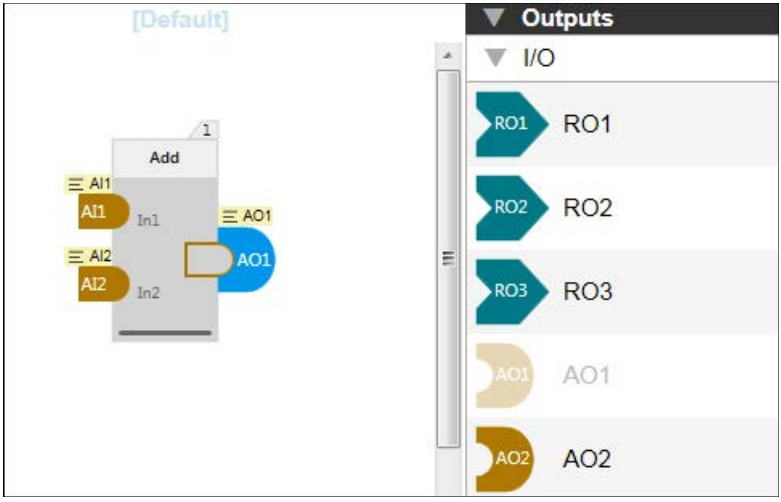


Figure 5. Outputs

For more information on output descriptions, refer *firmware manual(s)* in *List of related manuals*.

Creating a program

To create a program using function blocks, proceed as follows:

1. Drag-and-drop the desired function blocks to the canvas.

Figure 6. Function block

2. Drag-and-drop the desired inputs from the Input categories to the function block.

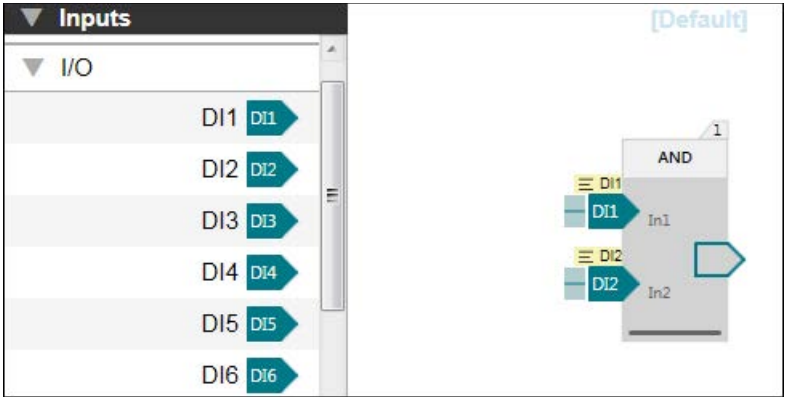


Figure 7. Adding inputs

3. Drag-and-drop the desired output from the Output categories to the function block.

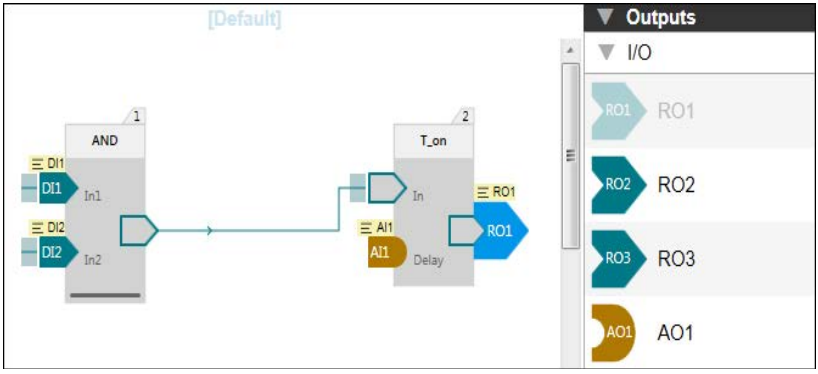


Figure 8. Adding output

Similarly, you can create the programs as desired by adding multiple function blocks using inputs and outputs.

Downloading the program to a drive

After creating a program in canvas, you can download the program to a drive.

1. Click **Download to drive**.

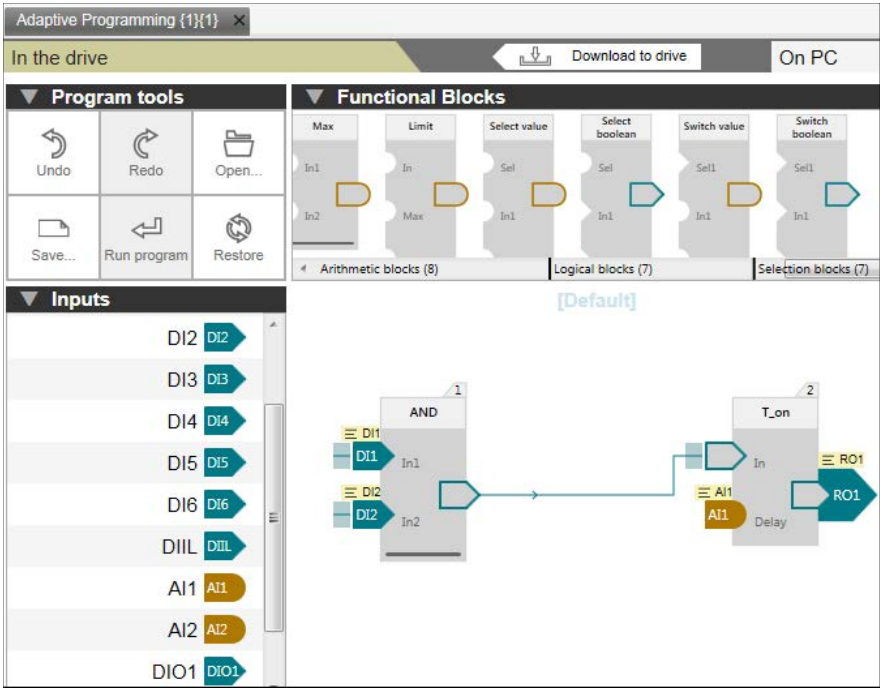


Figure 9. Downloading to drive

Program is downloaded to the drive.

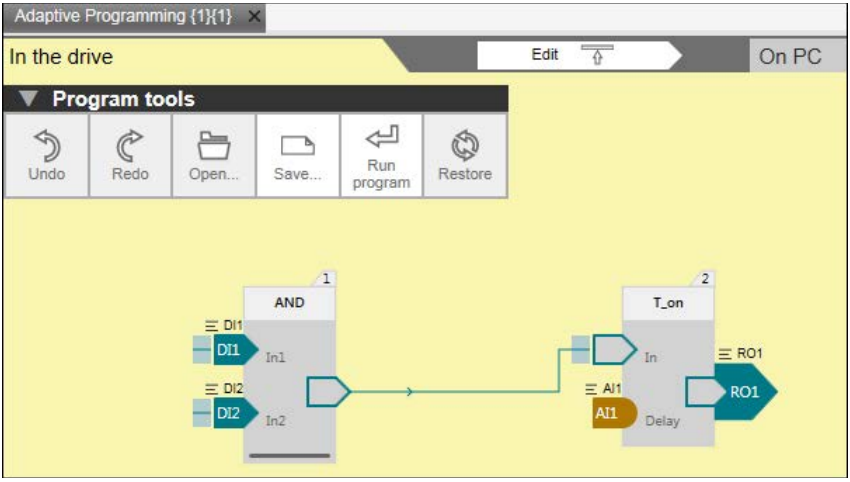


Figure 10. Program downloaded to a drive

After downloading to the drive, if you want to edit the program, click **Edit**. After editing, the program can be downloaded to the drive by clicking **Download to drive**.

2. In the Program tools, click **Run program** to start the program.
3. To stop the drive, click **Edit**.
4. Click **Save** to save the diagram to the PC.

4

Program elements

Contents of this chapter

This chapter describes system inputs, outputs and function blocks available in the master control program.

System inputs

The below mentioned system inputs are examples only.

Constants

Constants consists of Numerical and Boolean constant input values. These constant inputs can be reused in different blocks by changing their values.

For example: Numerical value and Boolean value.

I/O

Analog inputs

Analog inputs can be filtered, inverted or scaled with parameter configuration (i.e. not in Adaptive programming).

Analog inputs can be independently set as voltage or current input by a jumper. Each input can be filtered, inverted or scaled.

The drive can be set to perform an action if the value of an analog input moves out of a predefined range.

Digital inputs and outputs

Digital inputs and outputs can be set as either an input or an output.

Digital input/output DIO1 can be used as a frequency input, DIO2 as a frequency output.

For example: AI1, AI2, DI1, DI2, DIO1, DIO2 etc.

Actual values

Basic signals for monitoring the drive.

For example: Motor speed, Output frequency, Motor current and so on.

Status

Drive status word.

Example: Enabled, inhibited, Ready to start etc.

Data storage

Data storage parameters are reserved for data storage. These parameters are unconnected by default and can be used for linking, testing and commissioning purpose.

For example: Data storage 1 real32, Data storage 2 real32 etc.

For more information on Input descriptions, refer *firmware manual(s)* in *List of related manuals*.

System outputs

The below mentioned system outputs are examples only.

I/O

Analog outputs

Analog outputs can be filtered, inverted or scaled with parameter configuration (i.e not in Adaptive programming).

Relay outputs

The signal to be indicated by the outputs can be selected by parameters.

Digital inputs and outputs

Digital input/output DIO1 can be used as a frequency input, DIO2 as a frequency output.

For example: AO1, AO2, RO1, RO2, RO3, DIO1 and DIO2.

Start control

Operating mode

The two external control locations, EXT1 and EXT2, are available. The user can select the sources of the start and stop commands separately for each location.

Run enable

The source of the external run enable signal. If the run enable signal is switched off, the drive will not run.

Fault reset

The drive can automatically reset itself after overcurrent, overvoltage, undervoltage and external faults.

For example: Ext1/Ext2 selection, Run enable 1, Fault reset etc.

Speed control

The output of the speed reference selection block. The motor follows a speed reference given to the drive.

For example: Speed ref1, Speed ref2 and Speed additive 1.

Frequency control

The output of the frequency reference selection block. The motor follows a frequency reference given to the drive. Frequency control is only available in scalar motor control mode.

For example: Frequency ref1, Frequency ref2 etc.

Torque control

The output of the torque reference selection block. Motor torque follows a torque reference given to the drive.

For example: Torque ref1, Torque ref2 and Torque additive 2.

Limitations

Defines the source of maximum torque limit for the drive.

For example: Minimum torque 2 and Maximum torque 2.

Events

Defines the source of external events.

For example: External event 1, External event 2 etc.

Process PID

Selects the source that determines whether process PID parameter set is used.

For example: Set 1 setpoint 1, Set 1 feedback 1, Set 1 tracking mode etc.

For more information on output descriptions, refer *firmware manual(s)* in *List of related manuals*.

Function block specifications

You can adjust the number of inputs by dragging the bottom line in the function block.

Note: Function blocks which do not contain bottom line cannot be adjusted.

Abs

Calculates absolute value.

Output:

Name	Type	Default value
Out	Float	0

Input: 1

Name	Type	Default value	Function
In	Float	0	Block input

Block function

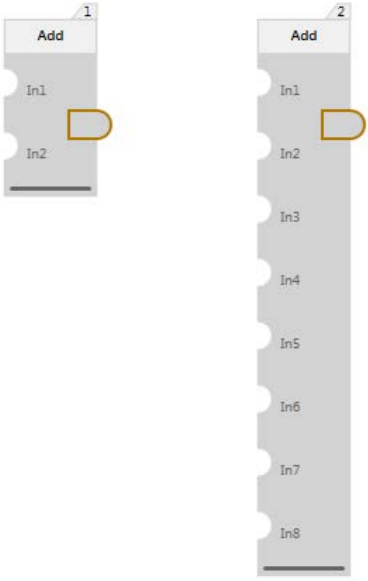
Block calculates absolute value of value in input *In*. Output = I *In* I.

Exceptional cases

Block input is not connected. Input has a default value.

Add

Adds **n** inputs and outputs result.



Output:

Name	Type	Default value
Out	Float	0

Inputs: 2-8

Default inputs: 2

Name	Type	Default value	Function
In1 - In8	Float	0	Provides values to add

Block function

Output = In1 + In2 +...+ In8

Exceptional cases

- Inputs which are not connected are added as default value.
- Overflow to positive side: output is limited to Max float.
- Overflow to the negative side: output is limited to negative Max float.
- Underflow: value 0 is kept at output.

AND

Performs logic *AND*.

Output

Name	Type	Default value
Out	Boolean	0

Inputs: 2-8

Default inputs: 2

Name	Type	Default value	Function
In1 - In8	Boolean	N/A	Block inputs

Block function

Function block performs logical conjunction operation with inputs.

Out = In1 & In2 & … & In8.

The truth table of AND operation is below. Example uses two inputs. Same logic can be applied to other inputs. Output is 1 (true) if and only if all inputs have value 1 (true).

In1	In2	Out
0	0	0
0	1	0
1	0	0
1	1	1

Exceptional cases

- Inputs which are not connected have no effect on the output.
- If some inputs are connected and others are not, only the connected inputs are evaluated.

Divide

Divides block inputs.

Output:

Name	Type	Default value
Out	Float	0

Inputs: 2

Name	Type	Default value	Function
Num	Float	0	Dividend
Denom	Float	0	Divisor

Block function

Output = In1 / In2

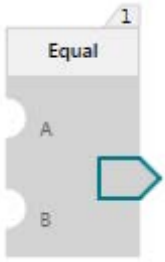
Dividing by zero will set block output to zero.

Exceptional cases

- Inputs which are not connected are assigned with default values.
- Overflow to positive side: output is limited to Max float.
- Overflow to the negative side: output is limited to negative Max float.
- Underflow: value 0 is kept at output.

Equal

Checks if values at inputs are equal.



Output

Name	Type	Default value
Out	Boolean	0

Inputs: 2

Name	Type	Default value	Function
A	Float	0	First comparison value
B	Float	0	Second comparison value

Block function

Block compares the whole number parts of numbers in A and B. Behavior of the block can be seen in table below.

Condition	Out
A and B are equal	1
A and B are not equal	0

Inputs are rounded before comparison. Only whole number part of the inputs are compared.

For example, if value 70.5 is in input, it will be compared as 71. If value 70.4 is in input it will be compared as 70. Rounding of negative numbers works as illustrated in the following example. -70.4 rounds to -70. -70.5 rounds to -71.

Exceptional cases

Inputs which are not connected will have a default value.

Filter

Filters input for a defined length of time and then outputs it.

Output:

Name	Type	Default value
Out	Float	0

Inputs: 2

Name	Type	Default value	Function
In	Float	0	Signal to be filtered
Time	Float	0	Filter time constant in seconds

Block function

This block is a single pole low - pass filter. Input signal *In* is filtered using provided time constant *Time*. The following equation is used for internal calculations.

```
Coefficient = TimeLevel / (TimeLevel + Time)
```

```
Out[i] = Coefficient * (In[i] – Out[i - 1]) + Out[i - 1]
```

Where:

Variable	Function
Out [i]	Current calculated output value
Out [i - 1]	Previous output value of the filter from previous time cycle
In [i]	Current input value
Timelevel	Value of timelevel that the program is running at.

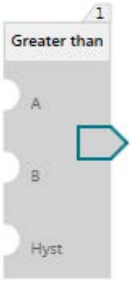
This function is a discrete model for single pole low - pass filter.

Exceptional cases

- Time constant *Time* < timelevel or negative constant is provided. Filter does not filter input signal. Input is written to output unaltered. Time constant is evaluated to 0.
- *In* is not connected - Input gets default value.
- *Time* constant is not connected - assumed to have default value.

Greater than

Comparison block. Compares values at its inputs to see if first value is greater than second. Comparison accuracy is set by the user.



Output

Name	Type	Default value
Out	Boolean	0

Inputs: 3

Name	Type	Default value	Function
A	Float	0	Provides first comparison value
B	Float	0	Provides second comparison value
Hyst	Float	0	Value B is subtracted

Block function

Takes two inputs to compare with one another, A and B, and a third input that manipulates input B.

First:

- If A > B, output is set to 1.

Second (if first is not true):

- If A < (B- Hyst) then output is reset to 0.

Third (if neither are true):

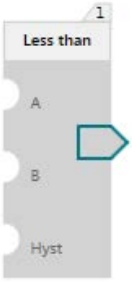
- Previous output value is kept at block output.

Exceptional cases

- When either A or B input is not connected then output is set to default value 0.
- A disconnected Hyst input has value 0.

Less than

Comparison block. Compares values at its inputs to see if first value is smaller than second. Comparison accuracy is set by the user.



Output

Name	Type	Default value
Out	Boolean	0

Inputs: 3

Name	Type	Default value	Function
A	Float	0	Provides first comparison value
B	Float	0	Provides second comparison value
Hyst	Float	0	Value that is added to B

Block function

Takes two inputs to compare with one another, A and B, and a third input that manipulates input B.

First

- If A < B, output is set to 1.

Second (if first isn't true)

- If A > (B+ Hyst) then output is reset to 0.

Third (if neither are true)

- Previous output value is kept at block output.

Exceptional cases

- When either A or B input is not connected then output is set to default value 0.
- A disconnected Hyst input has value 0.

Limit

Takes an input that is limited and outputs the value after limiting it.

Output:

Name	Type	Default value
Out	Float	0

Inputs: 3

Name	Type	Default value	Function
In	Float	0	Value to be limited.
Max	Float	3.4028235e+38	Maximum value *In* is limited
Min	Float	- 3.4028235e+38	Minimum value *In* is limited.

Block function

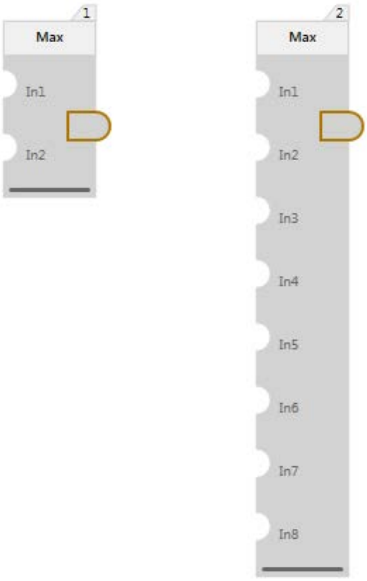
In is written to the output as long as it is within the value range of *Max* and *Min*. When *In* exceeds or falls below the respective limit values, it will first be capped to the appropriate limit value and then written to the output. *In* is evaluated first against *Max*. If *Max* is not limiting, then *In* is evaluated against *Min*.

Exceptional cases

- If *In* is not connected then the block output is zero.
- If *Max* or *Min* input is not connected, then the highest and lowest float values are set as the default values for *Max* or *Min*.

Max

Compares **n** inputs and outputs the largest input value.



Output:

Name	Type	Default value
Out	Float	0

Inputs: 2-8

Default inputs: 2

Name	Type	Default value	Function
In1 - In8	Float	0	Provides an input value to compare

Block function

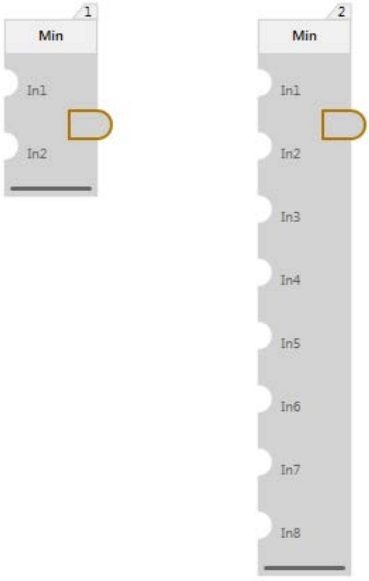
Compares all input values to determine the highest one and outputs it.

Exceptional cases

If some inputs are connected and other inputs are not connected, only the connected inputs are evaluated.

Min

Compares **n** inputs and outputs the smallest input value.



Output:

Name	Type	Default value
Out	Float	0

Inputs: 2-8

Default inputs: 2

Name	Type	Default value	Function
In - In8	Float	0	Provides an input value to be compared

Block function

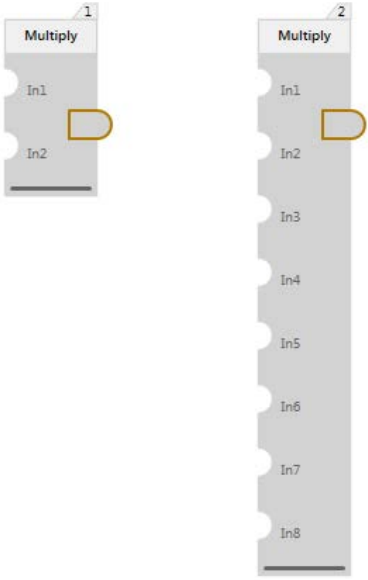
Compares all input values to determine the lowest one and outputs it.

Exceptional cases

If some inputs are connected and others are not connected, only the connected inputs are evaluated.

Multiply

Multiples **n** inputs and outputs the result.



Output:

Name	Type	Default value
Out	Float	0

Inputs: 2-8

Default inputs: 2

Name	Type	Default value	Function
In1 - In8	Float	N/A	Provides values for multiply block to perform multiplication

Block function

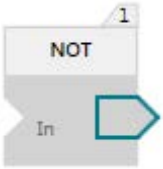
Out = In1 * In2 *...* In8

Exceptional cases

- Inputs which are not connected are not multiplied. If one input is connected, its value is at output.
- All inputs are not connected: output is assigned a default value.
- Overflow to positive side: output is limited to Max float.
- Overflow to the negative side: output is limited to negative Max float.
- Underflow: value 0 is kept at output.

NOT

Inverts value at input.



Output

Name	Type	Default value
Out	Boolean	1

Input: 1

Name	Type	Default value	Function
In	Boolean	0	Block input

Block function

Function block performs inversion.

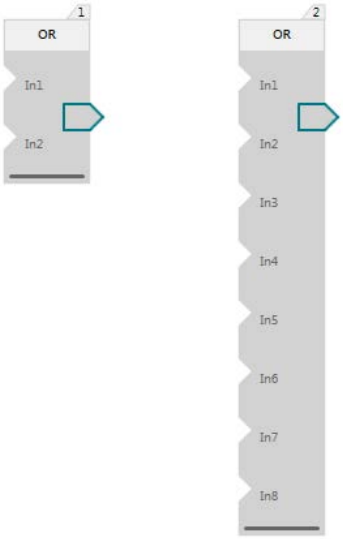
In	Out
0	1
1	0

Exceptional cases

In case a block input is not connected then its value is set to 0 by default.

OR

Performs logic *OR*.



Output

Name	Type	Default value
Out	Boolean	0

Inputs: 2-8

Default inputs: 2

Name	Type	Default value	Function
In1 - In8	Boolean	0	Block inputs

Block function

Function block performs logical or operation with inputs. Out = In1 v In2 v … v In8.

The truth table of OR operation is below. Example uses two inputs. Same logic can be applied to other inputs. Output has value 1 when one of the inputs have value 1. Output is 0 if and all inputs have value 0.

In0	In1	Out
0	0	0
0	1	1
1	0	1
1	1	1

Exceptional cases

If some inputs are connected and others are not, only the connected inputs are evaluated.

PI

PI controller.

Output:

Name	Type	Default value
Out	Float	0

Inputs: 8

Name	Type	Default value	Function
Setpoint	Float	0	Desired output value
Actual	Float	0	Actual output value
Gain	Float	0	Proportional gain (Kp)
Int time	Float	0	Integration time
Track	Boolean	0	Enables tracking mode
Track reference	Float	0	Output value in tracking mode
Min	Float	- 3.4028235e+38	Maximum output value
Max	Float	3.4028235e+38	Minimum output value

Block function

Calculates the P and I terms based on error, proportional gain and an integral coefficient. The sum of P and I is written to the output. Sets output to tracking reference value when tracking is enabled and limits the output when needed. In these cases, the I term value is maintained directly in reference to the tracking reference or limit values to provide smooth transfer/anti-windup. PI output continuous changing from track reference value when track is disabled. In the limitation, the value is evaluated first against *Max* limit. If *Max* is not limiting, then the value is evaluated against *Min* limit.

Exceptional cases

- In case a block input is not connected then its value is set to default value.
- When either *Setpoint*, *Actual* or *Gain* are not connected then output is set to 0. When *Track* is enabled and *Track reference* is not connected then output is set to 0.
- When Integration time input is not connected then integral component is reset and PI block functions as a P controller.
- When *Min* or *Max* is not connected, the default values of these inputs are used.

Ramp

Changes the output value to match the input value at a defined rate of change.

Output:

Name	Type	Default value
Out	Float	0

Inputs: 7

Name	Type	Default value	Function
In	Float	0	Reference value to ramp to output
Increase	Float	0	The amount of output increased per second
Decrease	Float	0	The amount of output decreased per second
Track	Boolean	0	Enables tracking mode
Track reference	Float	0	Output value in tracking mode
Max	Float	3.4028235e+38	Maximum value block *output will be limited*
Min	Float	- 3.4028235e+38	Minimum value block output will be limited

Block function

If output value does not equal input reference, then the output value starts changing towards the input value.

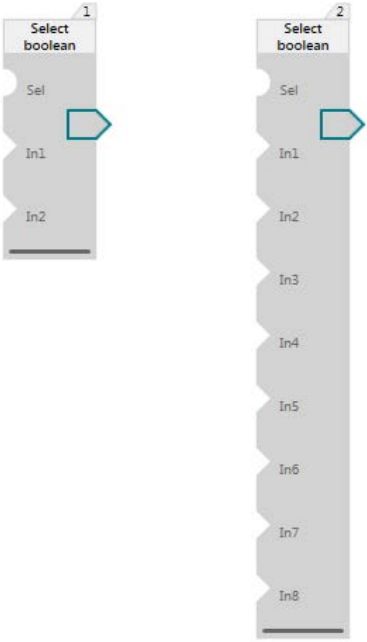
The amount of change per second is defined by the inputs for increasing and decreasing the output. Sets output to track reference value when track is enabled. Output is limited to maximum and minimum limit values. In the limitation, the output is evaluated first against *Max* limit. If *Max* is not limiting, then the output is evaluated against *Min* limit. Ramp output continues changing from tracking reference value when tracking is disabled.

Exceptional cases

- In case a block input is not connected, then its value is set to default value.
- In case, either maximum or minimum limit is disconnected, then their values will be defaulted to the highest and lowest value representable by a float.
- In case, *Increase* or *Decrease* input is disconnected then Output = *In* when trying to ramp with the disconnected input. If the other input is connected then ramping with it behaves as normal.
- In case, *In* input is disconnected then Output = 0.

Select boolean

Outputs the Boolean input value that is selected by the selector input.



Output

Name	Type	Default value
Out	Boolean	0

Inputs: 3-9

Default inputs: 3

Name	Type	Default value	Function
Sel	Float	0	Selects input value to connect to output
In1 - In8	Boolean	0	Provides selectable input value for the block.

Block function

This is a selector block that can have different input connected to output. Input to be connected is selected by *Sel* input.

When *Sel* = 1 then Out = *In1*, when *Sel* = 2 Out = *In2* etc.

When *Sel* = 8 Out = *In8*.

Allowable value range for *Sel* input is 1 <= *Sel* <= 8.

Exceptional cases

- When *Sel* input is out of its allowable range then Out = 0.
- Inputs which are not connected will have a default value.

Select value

Outputs the float input value that is selected by the selector input.

Output

Name	Type	Default value
Out	Float	0

Inputs: 3-9

Default inputs: 3

Name	Type	Default value	Function
Sel	Float	0	Selects input to be connect to output
In1 - In8	Float	0	Provides selectable input value for the block

Block function

This is a selector block that can have different input connected to output. Input to be connected is selected by Sel input.

When, *Sel* = 1 then Out = *In1*, and *Sel* = 2 then Out = *In2* and etc.

When, *Sel* = 8 then Out = *In8*.

Allowable value range for *Sel* input is 1 <= *Sel* <= 8.

Exceptional cases

- When *Sel* input is out of its allowable range then Output = 0.
- Inputs which are not connected will have a default value.

Set bits 0-7

Updates bits 0-7 of the input value.

Output

Name	Type	Default value
Out	Float	0

Inputs: 9

Name	Type	Default value	Function
In	Float	0	Value to be updated
Bit0	Boolean	N/A	Value of bit 0 (lowest)
Bit1	Boolean	N/A	Value of bit 1
Bit2	Boolean	N/A	Value of bit 2
Bit3	Boolean	N/A	Value of bit 3
Bit4	Boolean	N/A	Value of bit 4
Bit5	Boolean	N/A	Value of bit 5
Bit6	Boolean	N/A	Value of bit 6
Bit7	Boolean	N/A	Value of bit 7

Block function

Rounds the float input to closest integer and updates bits 0-7 of the integer value based on the boolean inputs Bit0-Bit7. Takes then the lowest 16 bits of the integer result and converts the value to float and writes it to output.

Exceptional cases

- When float input value is higher/lower than the range of unsigned 32 bit integer then the default value of the input is used. Bits 0-7 of the default value will be updated.
- If Boolean input is not connected, the value of that bit is not updated.

Set bits 8-15

Update bits 8-15 of the input value.

Output

Name	Type	Default value
Out	Float	0

Inputs: 9

Name	Type	Default value	Function
In	Float	0	Value to be updated
Bit8	Boolean	N/A	Value of bit 8
Bit9	Boolean	N/A	Value of bit 9
Bit10	Boolean	N/A	Value of bit 10
Bit11	Boolean	N/A	Value of bit 11
Bit12	Boolean	N/A	Value of bit 12
Bit13	Boolean	N/A	Value of bit 13
Bit14	Boolean	N/A	Value of bit 14
Bit15	Boolean	N/A	Value of bit 15

Block function

Rounds the float input to closest integer and updates bits 8-15 of the integer value based on the Boolean inputs Bit8-Bit15. Takes then the lowest 16 bits of the integer result and converts the value to float and writes it to output.

Exceptional cases

- When float input value is higher/lower than the range of unsigned 32 bit integer then the default value of the input is used. Bits 8-15 of the default value is updated.
- If Boolean input is not connected, the value of that bit is not updated.

SR

SR trigger is used to store Set value.

Output

Name	Type	Default value
Out	Boolean	0

Input: 2

Name	Type	Default value	Function
Set	Boolean	0	Set input
Reset	Boolean	0	Reset

Block function

This is SR latch. Output keeps it value once set by *Set* input. Value at output is reset to 0 when *Reset* = 1. Value at output depends on previous output value. See truth table.

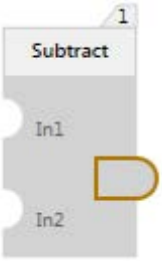
Previous Out	Reset	Set	Current Out
0	0	0	0
0	0	1	1
x	1	x	0
1	0	0	1
1	0	1	1

Exceptional cases

- If *Set* is not connected, it is assumed to have default value.
- If *Reset* is not connected, it is assumed to have default value.

Subtract

Performs subtract.



Output:

Name	Type	Default value
Out	Float	0

Inputs: 2

Name	Type	Default value	Function
In1	Float	0	Value to subtract from
In2	Float	0	Value to be subtracted

Block function

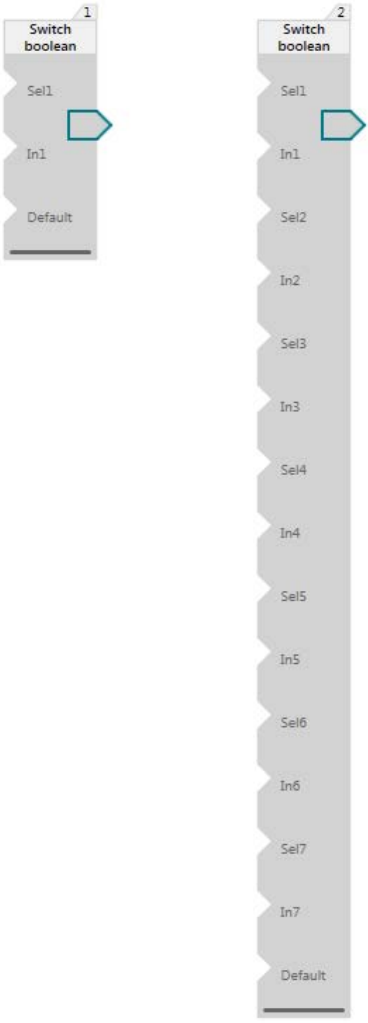
Output = In1 - In2

Exceptional cases

- In case both inputs are not connected, output has a default value.
- Inputs which are not connected are assigned default value.
- Overflow to positive side: output is limited to Max float.
- Overflow to the negative side: output is limited to negative Max float.
- Underflow: value 0 is kept at output

Switch boolean

Outputs the input Boolean value whose enable value is set first.



Output:

Name	Type	Default value
Out	Boolean	0

Inputs: 3-15

Default inputs: 3

Name	Type	Default value	Function
Sel1 - Sel7	Boolean	0	Selects/deselects input value.
In1 - In7	Boolean	0	Provides selectable input value for the block.
Default	Boolean	0	Default output when Sel is not active for any inputs.

Block function

The value written to the output is “In X” value whose “Sel X” is set first. If no “Sel X” is set then *Default* input is written to the output.

Example:

Multiple Sel inputs have value 1. Inputs are evaluated from top to bottom. In case of multiple *In*, Sel pairs *In1, Sel1* is checked first followed by *In2, Sel2* and etc. In case Multiple Sel inputs are 1 the first one will be connected to output. In this example, if both *Sel1* and *Sel 2* are 1 then *In1* is connected to output.

Exceptional cases

Inputs which are not connected will have a default value.

Switch value

Outputs the input float value whose enable value is set first.

Output:

Name	Type	Default value
Out	Float	0

Inputs: 3-15

Default inputs: 3

Name	Type	Default value	Function
Sel1 - Sel7	Boolean	0	Selects/deselects input value
In1 - In7	Float	0	Provides selectable input value for the block
Default	Float	0	Default, that is, connected to output when no Sel is 1

Block function

The value written to the output is “In X” value whose “Sel X” is set first. If no “Sel X” is set, then the *Default* input is written to the output.

Example:

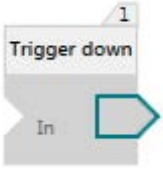
Multiple Sel inputs have value 1. Inputs are evaluated from top to bottom. In case of multiple *In*, Sel pairs *In1*, *Sel1* is checked first followed by *In2*, *Sel2* etc. In case Multiple Sel inputs are 1, the first one will be connected to output. In this example, if both *Sel1* and *Sel2* are 1 then *In1* is connected to output.

Exceptional cases

Inputs which are not connected will have a default value.

Trigger down

Falling edge detection.



Output

Name	Type	Default value
Out	Boolean	0

Input: 1

Name	Type	Default value	Function
In	Boolean	0	Block input

Block function

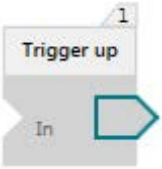
Function block performs falling edge detection. Output is 1 when input previous value is 1 and current value is 0. Otherwise output is 0.

Exceptional cases

- If input *In* is not connected, it will get the default value.
- If input *In* has value 0 at the first execution cycle of the block, the output of the block is set to 0.

Trigger up

Rising edge detection.



Output

Name	Type	Default value
Out	Boolean	0

Input: 1

Name	Type	Default value	Function
In	Boolean	0	Block input

Block function

Function block performs rising edge detection. Output is 1 when block input previous value is 0 and current value is 1. Otherwise output is 0.

Exceptional cases

- When input *In* is not connected, it will get the default value.
- If input *In* has value 1 at the first execution cycle of the block, the output of the block is set to 1.

T_off

Turns off the delay.

Output

Name	Type	Default value
Out	Boolean	0

Inputs: 2

Name	Type	Default value	Function
In	Boolean	0	Provides boolean value
Delay	Float	0	Provides time value to delay outputting 0

Block function

If the value of *In* is 1 then it is written to the output. If the value of *In* is 0 it is written to the output only after a time period is passed which is defined by *Delay*. *Delay* is limited to 2097152 seconds.

Exceptional cases

In case a block input is not connected, then its value is set to default value.

T_on

Turns on the delay.

Output

Name	Type	Default value
Out	Boolean	0

Inputs: 2

Name	Type	Default value	Function
In	Boolean	0	Provides boolean value.
Delay	Float	0	Provides time value to delay outputting 1.

Block function

If the value of *In* is 0 then it is written to the output. If the value of *In* is 1, it is written to the output only after a time period is passed which is defined by *Delay*. *Delay* is limited to 2097152 seconds.

Exceptional cases

In case a block input is not connected then its value is set to default value.

XOR

XOR inputs.

Output:

Name	Type	Default value
Out	Boolean	0

Inputs: 2

Name	Type	Default value	Function
In1	Boolean	0	Block input
In2	Boolean	0	Block input

Block function

Function block performs logical XOR operation with inputs.

The truth table of XOR operation:

In1	In2	Out
0	0	0
0	1	1
1	0	1
1	1	0

Output has value 1 when the inputs have different values, otherwise the output is 0.

Exceptional cases

In case a block input is not connected, the default value of the input is used in the operation.

Further information

Product and service inquiries

Address any inquiries about the product to your local ABB representative, quoting the type designation and serial number of the unit in question. A listing of ABB sales, support and service contacts can be found by navigating to www.abb.com/searchchannels.

Product training

For information on ABB product training, navigate to new.abb.com/service/training.

Providing feedback on ABB Drives manuals

Your comments on our manuals are welcome. Navigate to new.abb.com/drives/manuals-feedback-form.

Document library on the Internet

You can find manuals and other product documents in PDF format on the Internet at www.abb.com/drives/documents.

Contact us

www.abb.com/drives
www.abb.com/drivespartners

3AXD50000028574 Rev A (EN) 2015-06-30

Power and productivity
for a better world™